Alexander Garcia June 19, 2017

Assignment-4

1. Data: $[(0,0),(1,1),(2,3)]$

   (a) A full degree polynomial interpolation using the Lagrange basis is by definition

$$y = \sum_{i=1}^{n} y_i L_i(x_i)$$

   where

$$L_i = \prod_{k=1 \ k \neq i}^{n} \frac{x - x_k}{x_i - x_k}$$

   In this case, $n = 3$ and

   $x_1 = 0 \quad x_2 = 1 \quad x_3 = 2$
   $y_1 = 0 \quad y_2 = 1 \quad y_3 = 3$

   $y = 0L_1(x_1) + 1L_2(x_2) + 3L_3(x_3)$

$$L_2 = \frac{x-0}{1-0} * \frac{x-2}{1-2} = -x^2 + 2x$$

$$L_3 = \frac{x-0}{2-0} * \frac{x-1}{2-1} = \frac{x^2 - x}{2}$$

$$y = -x^2 + 2x + \frac{3}{2}(x^2 - x) = \frac{1}{2}(x^2 + x)$$

   Just to ensure the interpolant agrees with the data, a small MATLAB script was used to plot y with the points overlaid.

`holmes5_1.m` script

```
% check for Holmes 5.1

% define calculated interpolant
y = (1/2)*(x.^2 + x);
x = linspace(0,3);

% display results, including original data points
plot(x,y,0,0,'*',1,1,'*',2,3,'*');
legend('Interpolant','(x_1,y_1)','(x_2,y_2)','(x_3,y_3)','Location','Northwest');
```

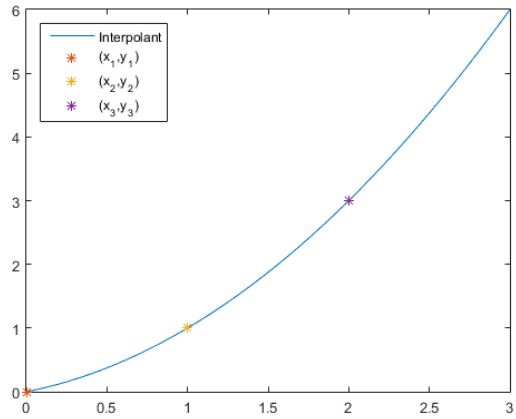

Figure 1: Graph of y and the data points

(b) Piecewise linear interpolation for this data set consists of 2 equations $S_1(x), S_2(x)$ which interpolate the data between $[x_1, x_2]$, $[x_2, x_3]$ respectively.
In general,

$$S_i(x) = a_i x + b_i$$

where

$$S_i(x_i) = y_i$$

From here, we get

$$a_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \qquad b_i = y_i - \frac{y_{i+1} - y_i}{x_{i+1} - x_i} * x_i$$

Then,

$$S_1(x) = \frac{1-0}{1-0}x + (0 - \frac{1-0}{1-0} * 0) = x \qquad x \in [0, 1]$$

and

$$S_2(x) = \frac{3-1}{2-1}x + (1 - \frac{3-1}{2-1} * 1) = 2x - 1 \qquad x \in [1, 2]$$

A simple script `holmes5_1_lin.m` was used to check the results of the interpolation.

`holmes5_1_lin.m`

```
% script to check part (b) of Holmes 5.1

% define domain
x1 = linspace(0,1);
x2 = linspace(1,2);

% define functions
S1 = x1;
S2 = 2*x2 - 1;
```

```
% display results, including original data points
plot(x1,S1,x2,S2,0,0,'*',1,1,'*',2,3,'*');
legend('S1','S2','(x_1,y_1)','(x_2,y_2)','(x_3,y_3)','Location','Northwest');
```

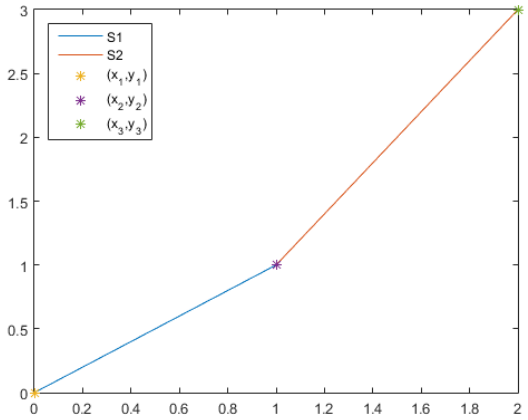

Figure 2: Graph of linear piecewise interpolant

(c) Natural cubic spline interpolation insists that the second derivative at both the first and last data points are zero.

Again, there are 2 equations $S_1(x), S_2(x)$ that interpolate the data between $[x_1, x_2]$, $[x_2, x_3]$ respectively. In general,

$$S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$

In order to satisfy the interpolation conditions, it must be enforced that $S_i(x_i) = y_i$. In addition, to ensure a smooth curve through the data, we insist that $S_i(x_i) = S_{i+1}(x_i)$, $S_i'(x_i) = S_{i+1}'(x_i)$, $S_i''(x_i) = S_{i+1}''(x_i)$

Given the initial conditions, we can construct a matrix to solve for the coefficients a_i, b_i, c_i, d_i. Each entry in matrix $\mathbf{A}$ is the value of x_i raised to the corresponding power.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 8 & 4 & 2 & 1 \\ 6 & 2 & 0 & 0 & -6 & -2 & 0 & 0 \\ 3 & 2 & 1 & 0 & -3 & -2 & -1 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 4 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \\ a_2 \\ b_2 \\ c_2 \\ d_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 3 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

A brief explanation of the contents of this matrix:

Rows 1-4: Interpolation conditions; $S_i(x_i) = y_i$
Rows 5-6: "Smoothness" conditions; agreeing first and second derivatives
Rows 7-8: Natural spline conditions; zero end point second derivatives

Solving this system using the backslash command in MATLAB yields

$$\mathbf{c} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & -3 & 4 & -1 \end{bmatrix}^T$$

Using these results, we have

$$S_1(x) = x$$

and

$$S_2(x) = x^3 - 3x^2 + 4x - 1$$

Below is the script used to both calculate and plot the resulting splines.

`holmes5_1_cubic.m`

```
% Script to check part (c) of Holmes5.1

% define coefficient matrix A
A = [ 0 0 0 1 0 0 0 0;
      1 1 1 1 0 0 0 0;
      0 0 0 0 1 1 1 1;
      0 0 0 0 8 4 2 1;
      6 2 0 0 -6 -2 0 0;
      3 2 1 0 -3 -2 -1 0;
      0 2 0 0 0 0 0 0;
      0 0 0 0 12 4 0 0; ];

% define resultant vector b
b = [0 1 1 3 0 0 0 0]';

% calculate vector of coefficients
c = A\b;

% define domains
x1 = linspace(0,1);
x2 = linspace(1,2);

% define splines from c
S1 = c(1)*x1.^3 + c(2)*x1.^2 + c(3)*x1 + c(4);
S2 = c(5)*x2.^3 + c(6)*x2.^2 + c(7)*x2 + c(8);

% display results, including original data points
plot(x1,S1,x2,S2,0,0,'*',1,1,'*',2,3,'*');
legend('S1','S2','(x_1,y_1)','(x_2,y_2)','(x_3,y_3)','Location','Northwest');
```

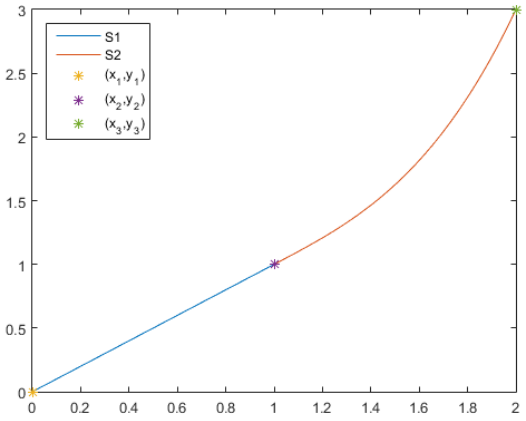

Figure 3: Graph of cubic piecewise interpolant

2. Function to interpolate: $f(x) = sinx$

   (a) Piecewise linear interpolation
   The linear spline passing through $\frac{\pi}{8}$ would be $S_1(x)$.

$$S_1(x) = \frac{y_2 - y_1}{x_2 - x_1}x + (y_1 - \frac{y_2 - y_1}{x_2 - x_1})x_1 \qquad x \in [0, \frac{\pi}{4}]$$

$$S_1(x) = \frac{\sqrt{2}/2 - 0}{\pi/4 - 0}x + (\sqrt{2}/2 - \frac{\sqrt{2}/2 - 0}{\pi/4 - 0})0$$

$$= \frac{\sqrt{2}/2}{\pi/4}x$$

   A MATLAB script `holmes5_5a.m` was used to calculate the estimated value of $f(\pi/8)$, and plot the interpolant and the original function.

$$S_1(\pi/8) = 0.35355$$
$$|S_1(\pi/8) - sin(\pi/8)| = 0.02913$$

`holmes5_5a.m`

```
% check linear interpolation of f(x) = sinx
clc;
% function definition
f = sin(x);
```

```
% calculated interpolant definition
s1 = ((sqrt(2)/2)/(pi/4))*x;

% display original plot, interpolant, and data points
x = linspace(0,pi/4);
plot(x,f,x,s1,0,0,'*',pi/4,sqrt(2)/2,'*');
legend('f(x)','S_1(x)','(x_1,y_1)','(x_2,y_2)','Location','Northwest');

% calculate error in interpolant at x = \pi/8
est = ((sqrt(2)/2)/(pi/4))*(pi/8);

err = est - sin(pi/8);
ea = abs(err);
fprintf('Estimated value of f(pi/8): %.5f\n', est);
fprintf('Absolute error in evaluating S_1(pi/8): %.5f\n', ea);
```

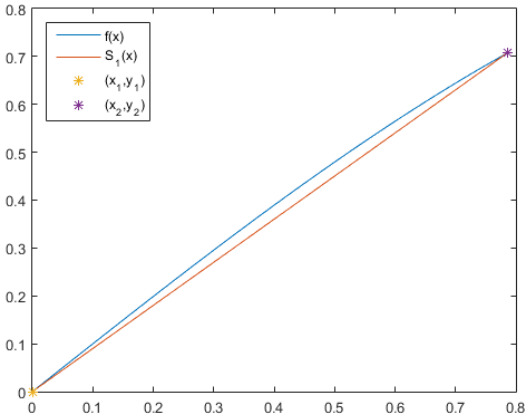

Figure 4: Graph of $f(x)$ and $S_1(x)$

(b) Full degree interpolation using the Lagrange basis

The Lagrange interpolant in this case uses $n = 3$, with data points

$x_1 = 0 \quad x_2 = \frac{\pi}{4} \quad x_3 = \frac{\pi}{2}$

$y_1 = 0 \quad y_2 = \frac{\sqrt{2}}{2} \quad y_3 = 1$

$y = 0L_1(x_1) + (\sqrt{2}/2)L_2(x_2) + 1L_3(x_3)$

$$L_2(x_2) = \frac{x-0}{\pi/4-0} * \frac{x-\pi/2}{\pi/4-\pi/2} = -\frac{16x^2-8\pi x}{\pi^2}$$

$$L_3(x_3) = \frac{x-0}{\pi/2-0} * \frac{x-\pi/4}{\pi/2-\pi/4} = \frac{8x^2-2\pi x}{\pi^2}$$

$$y = -\frac{\sqrt{2}}{2} * \frac{16x^2 - 8\pi x}{\pi^2} + \frac{8x^2 - 2\pi x}{\pi^2} = \frac{(8 - 8\sqrt{2})x^2 + (4\sqrt{2}\pi - 2\pi)x}{\pi^2}$$

Again, MATLAB was used to verify the result, as well as calculate the error.

$$y(\pi/8) = 0.40533$$
$$|y(\pi/8) - sin(\pi/8)| = 0.02265$$

holmes5_5b.m

```
% check lagrange interpolation of f(x) = sinx
clc;
% function definition
f = sin(x);

% calculated interpolant definition
s1 = ((8-8*sqrt(2))*x.^2+(4*sqrt(2)*pi-2*pi)*x)/pi^2;

% display original plot, interpolant, and data points
x = linspace(0,pi/2);
plot(x,f,x,s1,0,0,'*',pi/4,sqrt(2)/2,'*',pi/2,1,'*');
legend('f(x)','y','(x_1,y_1)','(x_2,y_2)','(x_3,y_3)','Location','Northwest');

% calculate error in interpolant at x = \pi/8
est = ((8-8*sqrt(2))*(pi/8)^2+(4*sqrt(2)*pi-2*pi)*(pi/8))/pi^2;

err = est - sin(pi/8);
ea = abs(err);
fprintf('Estimated value of f(pi/8): %.5f\n', est);
fprintf('Absolute error in evaluating y(pi/8): %.5f\n', ea);
```

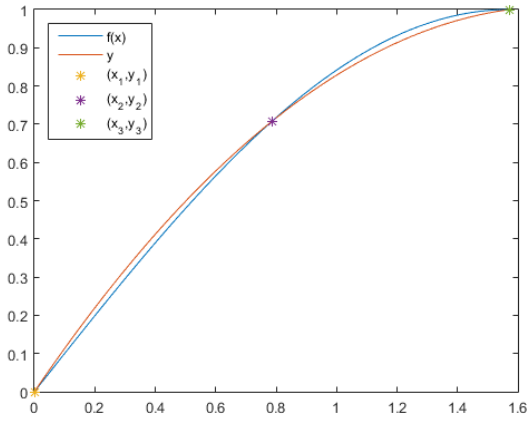

Figure 5: Graph of $f(x)$ and full degree interpolating polynomial y

(c) Natural cubic spline interpolation

Rows 1-4: Interpolation conditions; $S_i(x_i) = y_i$
Rows 5-6: "Smoothness" conditions; agreeing first and second derivatives
Rows 7-8: Natural spline conditions; zero end point second derivatives

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \pi^3/64 & \pi^2/16 & \pi/4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \pi^3/64 & \pi^2/16 & \pi/4 & 1 \\ 0 & 0 & 0 & 0 & \pi^3/8 & \pi^2/4 & \pi/2 & 1 \\ 3\pi^2/16 & \pi/2 & 1 & 0 & -3\pi^2/16 & -\pi/2 & -1 & 0 \\ 3\pi/2 & 2 & 0 & 0 & -3\pi/2 & -2 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3\pi & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \\ a_2 \\ b_2 \\ c_2 \\ d_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \sqrt{2}/2 \\ \sqrt{2}/2 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

A MATLAB script was used to calculate the resulting coefficients, the error of the interpolating spline, as well as plot the spline and $f(x)$. The resulting splines are

$$\begin{aligned} S_1(x) &= -0.21374x^3 + 1.03216x & x \in [0, \pi/4] \\ S_2(x) &= 0.21374x^3 - 1.00725x^2 + 1.82325x - 0.20711 & x \in [\pi/4, \pi/2] \end{aligned}$$

$$S_1(\pi/8) = 0.39239$$

$$|S_1(\pi/8) - sin(\pi/8)| = 0.00970$$

`holmes5_5c.m` script

```
% Script to check part (c) of Holmes5.5
clc;

% function definition
f = sin(x);

% define coefficient matrix A
A = [ 0 0 0 1 0 0 0 0;
    pi^3/64 pi^2/16 pi/4 1 0 0 0 0;
    0 0 0 0 pi^3/64 pi^2/16 pi/4 1;
    0 0 0 0 pi^3/8 pi^2/4 pi/2 1;
    3*pi^2/16 pi/2 1 0 -3*pi^2/16 -pi/2 -1 0;
    3*pi/2 2 0 0 -3*pi/2 -2 0 0;
    0 2 0 0 0 0 0 0;
    0 0 0 0 3*pi 2 0 0; ];

% define resultant vector b
b = [0 sqrt(2)/2 sqrt(2)/2 1 0 0 0 0]';

% calculate vector of coefficients
c = A\b;

fprintf('S1 = %.5fx^3 + %.5fx^2 + %.5fx + %.5f\n',c(1),c(2),c(3),c(4));
fprintf('S2 = %.5fx^3 + %.5fx^2 + %.5fx + %.5f\n',c(5),c(6),c(7),c(8));

% define domains
x = linspace(0,pi/2);
x1 = linspace(0,pi/4);
x2 = linspace(pi/4,pi/2);

% define splines from c
S1 = c(1)*x1.^3 + c(2)*x1.^2 + c(3)*x1 + c(4);
S2 = c(5)*x2.^3 + c(6)*x2.^2 + c(7)*x2 + c(8);

% display results, including original data points
plot(x,f,x1,S1,x2,S2,0,0,'*',pi/4,sqrt(2)/2,'*',pi/2,1,'*');
legend('f','S1','S2','(x_1,y_1)','(x_2,y_2)','(x_3,y_3)','Location','Northwest');

% calculate error in interpolant at x = \pi/8
est = c(1)*(pi/8)^3 + c(2)*(pi/8)^2 + c(3)*(pi/8) + c(4);

err = est - sin(pi/8);
ea = abs(err);
fprintf('Estimated value of f(pi/8): %.5f\n', est);
fprintf('Absolute error in evaluating S_1(pi/8): %.5f\n', ea);
```

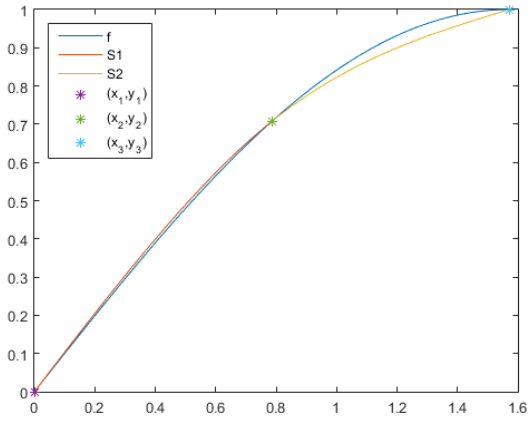

Figure 6: Graph of $f(x)$ and natural cubic splines $S_1(x), S_2(x)$

(d) Clamped cubic spline interpolation

Rows 1-4: Interpolation conditions; $S_i(x_i) = y_i$
Rows 5-6: "Smoothness" conditions; agreeing first and second derivatives
Rows 7-8: Clamped spline conditions; fixed end point first derivatives

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ \pi^3/64 & \pi^2/16 & \pi/4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \pi^3/64 & \pi^2/16 & \pi/4 & 1 \\ 0 & 0 & 0 & 0 & \pi^3/8 & \pi^2/4 & \pi/2 & 1 \\ 3\pi^2/16 & \pi/2 & 1 & 0 & -3\pi^2/16 & -\pi/2 & -1 & 0 \\ 3\pi/2 & 2 & 0 & 0 & -3\pi/2 & -2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3\pi^2/4 & \pi & 1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ b_1 \\ c_1 \\ d_1 \\ a_2 \\ b_2 \\ c_2 \\ d_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \sqrt{2}/2 \\ \sqrt{2}/2 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Once again, the coefficients were calculated by a MATLAB script, as well as the error, and a plot of the spline and function.

$$\begin{aligned} S_1(x) &= -0.34178x^3 + 0.14151x^2 + x && x \in [0, \pi/4] \\ S_2(x) &= 0.49355x^3 - 1.82668x^2 + 2.54582x - 0.40469 && x \in [\pi/4, \pi/2] \end{aligned}$$

$$S_1(\pi/8) = 0.39382$$

$$|S_1(\pi/8) - sin(\pi/8)| = 0.01114$$

`holmes5_5d.m` script

```
% Script to check part (d) of Holmes5.5
clc;

% function definition
f = sin(x);

% define coefficient matrix A
A = [ 0 0 0 1 0 0 0 0;
    pi^3/64 pi^2/16 pi/4 1 0 0 0 0;
    0 0 0 0 pi^3/64 pi^2/16 pi/4 1;
    0 0 0 0 pi^3/8 pi^2/4 pi/2 1;
    3*pi^2/16 pi/2 1 0 -3*pi^2/16 -pi/2 -1 0;
    3*pi/2 2 0 0 -3*pi/2 -2 0 0;
    0 0 1 0 0 0 0 0;
    0 0 0 0 3*pi^2/4 2 0 0; ];

% define resultant vector b
b = [0 sqrt(2)/2 sqrt(2)/2 1 0 0 1 0]';

% calculate vector of coefficients
c = A\b;

fprintf('S1 = %.5fx^3 + %.5fx^2 + %.5fx + %.5f\n',c(1),c(2),c(3),c(4));
fprintf('S2 = %.5fx^3 + %.5fx^2 + %.5fx + %.5f\n',c(5),c(6),c(7),c(8));

% define domains
x = linspace(0,pi/2);
x1 = linspace(0,pi/4);
x2 = linspace(pi/4,pi/2);

% define splines from c
S1 = c(1)*x1.^3 + c(2)*x1.^2 + c(3)*x1 + c(4);
S2 = c(5)*x2.^3 + c(6)*x2.^2 + c(7)*x2 + c(8);

% display results, including original data points
plot(x,f,x1,S1,x2,S2,0,0,'*',pi/4,sqrt(2)/2,'*',pi/2,1,'*');
legend('f','S1','S2','(x_1,y_1)','(x_2,y_2)','(x_3,y_3)','Location','Northwest');

% calculate error in interpolant at x = \pi/8
est = c(1)*(pi/8)^3 + c(2)*(pi/8)^2 + c(3)*(pi/8) + c(4);

err = est - sin(pi/8);
ea = abs(err);
fprintf('Estimated value of f(pi/8): %.5f\n', est);
fprintf('Absolute error in evaluating S_1(pi/8): %.5f\n', ea);
```

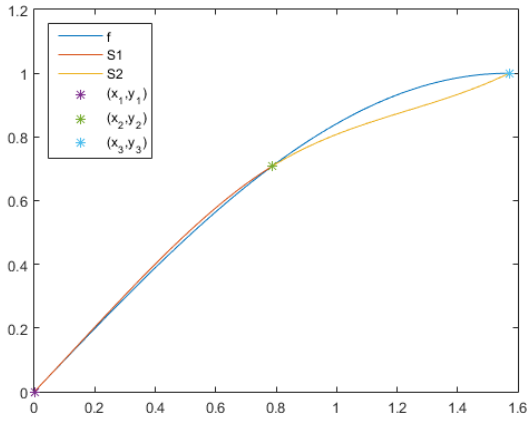

Figure 7: Graph of $f(x)$ and clamped cubic splines $S_1(x), S_2(x)$

(e) Chebyshev interpolation

The function needed to determine the nodes for full-degree interpolation are given by the zeros of the Chebyshev polynomial $T_n(x) = cosn\theta$, which occur at $n\theta = \frac{(2k-1)\pi}{2n}$, where $k = 1 : n$. However, this only applies to the interval $[-1, 1]$, and therefore must be expanded to our interval $[0, \pi/2]$. When generalized, the zeros lie at $\frac{b+a}{2} - \frac{b-a}{2}cos(\frac{(2k-1)\pi}{2n})$, again with $k = 1 : n$. In our case then, $n = 3$, $b = \pi/2$, $a = 0$, and the zeros are

$$\begin{aligned} k = 1: &\quad x_1 = \tfrac{\pi}{4} - \tfrac{\pi}{4}cos(\tfrac{\pi}{6}) = 0.1052 \\ k = 2: &\quad x_2 = \tfrac{\pi}{4} - \tfrac{\pi}{4}cos(\tfrac{\pi}{2}) = \tfrac{\pi}{4} \\ k = 3: &\quad x_3 = \tfrac{\pi}{4} - \tfrac{\pi}{4}cos(\tfrac{5\pi}{6}) = 1.4656 \end{aligned}$$

We then use these nodes, to generate a full degree polynomial interpolant, in this case using the Lagrange basis.

$$y = y_1L_1(x_1) + y_2L_2(x_2) + y_3L_3(x_3)$$

$$\begin{aligned} L_1(x_1) &= \frac{x - \pi/4}{0.1052 - \pi/4} * \frac{x - 1.4656}{0.1052 - 1.4656} \\ L_2(x_2) &= \frac{x - 0.1052}{\pi/4 - 0.1052} * \frac{x - 1.4656}{\pi/4 - 1.4656} \\ L_3(x_3) &= \frac{x - 0.1052}{1.4656 - 0.1052} * \frac{x - \pi/4}{1.4656 - \pi/4} \end{aligned}$$

$$y(x) = sin(x_1)L_1(x_1) + sin(x_2)L_2(x_2) + sin(x_3)L_3(x_3)$$

The function estimate, as well as the error, were calculated through a MATLAB script. The script also plots the interpolant and the original function.

$$y(\pi/8) = 0.39790$$

$$|y(\pi/8) - sin(\pi/8)| = 0.01521$$

`holmes5_5e.m` script

```
% script to calculate full degree interpolant based on
% ideal node spacing
clc;

% define domain
xx = linspace(0,pi/2);

% define original function
f = sin(xx);

% calculate ideal nodes
x1 = (pi/4)-(pi/4)*cos(pi/6);
x2 = (pi/4)-(pi/4)*cos(pi/2);
x3 = (pi/4)-(pi/4)*cos(5*pi/6);

% calculate lagrange basis polynomial based on ideal nodes
L1 = (xx-x2)./(x1-x2) .* (xx-x3)./(x1-x3);
L2 = (xx-x1)./(x2-x1) .* (xx-x3)./(x2-x3);
L3 = (xx-x1)./(x3-x1) .* (xx-x2)./(x3-x2);

y = sin(x1).*L1 + sin(x2).*L2 + sin(x3).*L3;

% display interpolant, original function, and data points
plot(xx,y,xx,f,x1,sin(x1),'*',x2,sin(x2),'*',x3,sin(x3),'*');
legend('y','f','(x_1,y_1)','(x_2,y_2)','(x_3,y_3)','Location','Northwest');

% calculate error in interpolant at y(\pi/8)
xx = pi/8;

L1 = (xx-x2)./(x1-x2) .* (xx-x3)./(x1-x3);
L2 = (xx-x1)./(x2-x1) .* (xx-x3)./(x2-x3);
L3 = (xx-x1)./(x3-x1) .* (xx-x2)./(x3-x2);

est = sin(x1).*L1 + sin(x2).*L2 + sin(x3).*L3;

err = est-sin(pi/8);
ea = abs(err);

fprintf('Estimated value of f(pi/8): %.5f\n',est);
fprintf('Absolute error in evaluating y(pi/8): %.5f\n',ea);
```

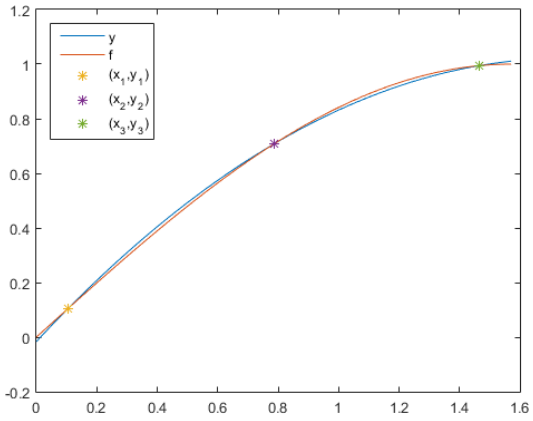

Figure 8: Graph of $f(x)$ and the Chebyshev polynomial $y(x)$

3. Function to interpolate: $y(x) = log_{10}x$, where $x \in [0, 10]$

   (a) When using a piecewise linear interpolant $p_1(x)$ to approximate a function $f(x)$, the error for a given segment from x_i to x_{i+1} is given as

$$|f(x) - p_1(x)| \leq \frac{1}{2} max|f''(x)| * max|w(x)|$$

   where

$$w(x) = (x - x_i)(x - x_{i+1})$$

   We can then define the distance between x_i and x_{i+1} as h. If we use this definition over the first two points in the interpolation $x_1 = 0$, $x_2 = h$, then we can see that $w(x) = (x)(x-h)$, and $w(x)$ will reach its maximum magnitude at $x = h/2$, a value of $\frac{h^2}{4}$. This result will hold for not only the first interval, but all intervals used for interpolation. Therefore,

$$|f(x) - p_1(x)| \leq \frac{h^2}{8} max|f''(x)|$$

   We are given the maximum error as 10^{-6} and $f(x)$. The only unknown then is a node spacing h that will satisfy the inequality.

$$10^{-6} = \frac{h^2}{8} max|-\frac{1}{x^2 log(10)}|$$

   The maximum of $f''(x)$ can be found by the zeros of $f'''(x)$

$$f'''(x) = \frac{2}{x^3 log(10)}$$

   This function has no zeros, so the maximum will occur at one of the ends of the interval $[1, 10]$.

$$|f''(1)| = \frac{1}{log(10)} \approx 0.434294$$

$$|f''(10)| = \frac{1}{100log(10)} \approx 0.00434294$$

We now have

$$10^{-6} = \frac{h^2}{8} * \frac{1}{log(10)}$$

Solving for h, we get $h = \frac{1}{250}\sqrt{\frac{log(10)}{2}}$, which is the largest possible node spacing to ensure an error of at most 10^{-6}. The number of data points then needed to cover the interval $[1, 10]$ is $9/h = 2250\sqrt{\frac{2}{log(10}}$. Rounding this up, we get

$$n = 2097$$

(b) A process similar to that of piecewise linear error estimation can be used to estimate the error in a clamped cubic spline interpolant. The maximum error of the cubic spline interpolant is given as

$$|f(x) - S(x)| \leq \frac{3h^4}{384} max|f^{(iv)}(x)|$$

Here,

$$|f^{(iv)}(x)| = \frac{6}{x^4 log(10)}$$

which takes its maximum value at $x = 1$

$$|f^{(iv)}(1)| = \frac{6}{log(10)}$$

We now have everything we need to solve for h.

$$10^{-6} = \frac{3h^4}{384}\frac{6}{log(10)}$$

$$h \approx 0.033275$$

The minimum number of nodes for the interval $[1, 10]$ is then

$$\lceil 9/h \rceil = 271$$

(c) The error estimate for a polynomial interpolant given ideal node spacing is

$$|f(x) - p_n(x)| \leq \frac{1}{2^n(n+1)!}(\frac{b-a}{2})^{n+1}||f^{(n+1)}||_\infty$$

where

$$||f^{(n+1)}(x)||_\infty = max|f^{(n+1)}(x)|,\ x \in [a,b]$$

and n is the number of data points. This error is not quite as easy to solve, as n is involved in which derivative of f is being used, but this can be avoided by realizing the general expression for the derivatives of $log_{10}(x)$.

$$||f^{(n+1)}(x)||_\infty = \frac{n!}{x^{n+1}log(10)}$$

where the maximum will always occur at $x = 1$. We now look for values of n such that

$$10^{-6} = \frac{||f^{(n+1)}(1)||_\infty}{2^n(n+1)!}$$

as this is the smallest possible n that will ensure the error is below the tolerance. With some help from the computer, we can now solve for n.

$$\lceil n \rceil = 15$$

4. Function to interpolate:

$$g(x) = \begin{cases} 2 + 3x^2 + \alpha x^3 & -1 \le x \le 0 \\ 2 + \beta x^2 - x^3 & 0 \le x \le 1 \end{cases}$$

(a) In order for $g(x)$ to be a cubic spline, it must be twice differentiable at $x = 0$.

$$g'(x) = \begin{cases} 6x + 3\alpha x^2 & -1 \le x \le 0 \\ 2\beta x - 3x^2 & 0 \le x \le 1 \end{cases}$$

$$g''(x) = \begin{cases} 6 + 6\alpha x & -1 \le x \le 0 \\ 2\beta - 6x & 0 \le x \le 1 \end{cases}$$

continuous $g(0)$	$\rightarrow$	$2 = 2$
continuous $g'(0)$	$\rightarrow$	$0 = 0$
continuous $g''(0)$	$\rightarrow$	$2\beta = 6$

Based on these results, $\beta = 3$, but α is arbitrary. The two pieces of the function will always be not only equal, but will have the same first and second derivatives at $x = 0$ as long as $\beta = 3$. We simply choose $\alpha = 1$ to keep things simple. The cubic spline through these points is therefore

$$g(x) = \begin{cases} 2 + 3x^2 + x^3 & -1 \le x \le 0 \\ 2 + 3x^2 - x^3 & 0 \le x \le 1 \end{cases}$$

(b) The only data point that was concerning here was the point $x = 0$. As there were only two cubic functions making up the spline, there is only one point at which they must agree. In this case, that point is $(0, 2)$.

(c) In order for $g(x)$ to be a natural cubic spline, the second derivatives at each endpoint must be zero. In this case, this means

$$g''(-1) = g''(1) = 0$$

In other words,

$$2 + 3(-1)^2 + \alpha(-1)^3 = 0$$
$$2 + \beta(1)^2 - (-1)^3 = 0$$

$g(x)$ must still be twice differentiable at $x = 0$, so we have added two constraints to the conditions in part (a).

$g''(-1) = 0$	$\rightarrow$	$\alpha = 5$
$g''(1) = 0$	$\rightarrow$	$\beta = -3$
continuous $g(0)$	$\rightarrow$	$2 = 2$
continuous $g'(0)$	$\rightarrow$	$0 = 0$
continuous $g''(0)$	$\rightarrow$	$2\beta = 6$

We have reached a contradiction, as the definition of a natural spline demands that $\beta = -3$, but the conditions for a cubic spline demand that $\beta = 3$. Therefore, there are no values of α and β that make $g(x)$ a natural cubic spline.

(d) In order for $g(x)$ to be a clapmed cubic spline, $g'(-1) = f'(-1),\ g'(1) = f'(1)$. However, we are only given the function $g(x)$, and not the function that is interpolating. Therefore, we do not know $f'(-1),\ f'(1)$, and are left with the same results as part (a). $\beta = 3$, and α is arbitrary.

5. (a) A fairly straightforward MATLAB script was used to calculate the interpolating cubic spline. The script, `holmes5_16a.m`, stores all given data, and uses the built in `spline` routine to calculate an interpolant that passes through all data points. Next, the `ppval` routine is used to generate a continuous plot of the spline over the entire domain. Included below are both `holmes5_16a.m` and the graph of the airfoil generated by the script.

`holmes5_16a.m`

```
% model an airfoil using piecewise cubic splines
clc;

% define domain
d = linspace(0,1);

% given data points
x = [0 0.005 0.0075 0.0125 0.025 0.05 0.1...
```

```
    0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0];

yu = [0  0.0102  0.0134  0.017  0.025  0.0376  0.0563...
    0.0812  0.0962  0.1035  0.1033  0.095  0.0802  0.0597  0.034  0];

yl = [0  -0.0052  -0.0064  -0.0063  -0.0064  -0.006  -0.0045  -0.0016  0.001...
    0.0036  0.007  0.0121  0.017  0.0199  0.0178  0];

% generate upper and lower splines
uspline = spline(x,yu);
lspline = spline(x,yl);

% draw interpolants
zu = ppval(uspline,d);
zl = ppval(lspline,d);

plot(x,yu,'ro',x,yl,'o',d,zu,d,zl);
legend('Upper spline data', 'Lower spline data', 'Upper interpolant',...
    'Lower interpolant');
```

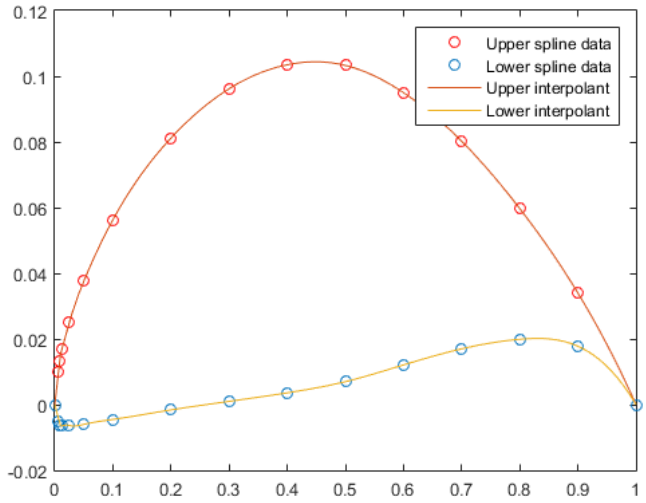

Figure 9: Graph of interpolating cubic splines and given discrete data

(b) Another MATLAB script was used to calculate the full-degree polynomial interpolation of the airfoil. However, given the naturally wiggly nature of these interpolants, the resulting graph was completely unrecognizable. The initial graph, Figure 10, is the automatically scaled graph of the polynomial from $[0, 1]$. Figure 11 is a zoomed in version, such that the initial data points are still visible.

`holmes5_16b.m`

```
% model an airfoil using full degree polynomials
clc;

% define domain
d = linspace(0,1);

% given data points
x = [0 0.005 0.0075 0.0125 0.025 0.05 0.1...
    0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1.0];

yu = [0 0.0102 0.0134 0.017 0.025 0.0376 0.0563...
    0.0812 0.0962 0.1035 0.1033 0.095 0.0802 0.0597 0.034 0];

yl = [0 -0.0052 -0.0064 -0.0063 -0.0064 -0.006 -0.0045 -0.0016 0.001...
    0.0036 0.007 0.0121 0.017 0.0199 0.0178 0];

% generate upper and lower splines
upoly = polyfit(x,yu,16);
lpoly = polyfit(x,yl,16);

% draw interpolants
zu = polyval(upoly,d);
zl = polyval(lpoly,d);

plot(x,yu,'ro',x,yl,'o',d,zu,d,zl);
legend('Upper airfoil data', 'Lower airfoil data', 'Upper interpolant',...
    'Lower interpolant');

% redefine y-axis limits in order to see original airfoil
axis([0 1 -0.01 0.15]);
```

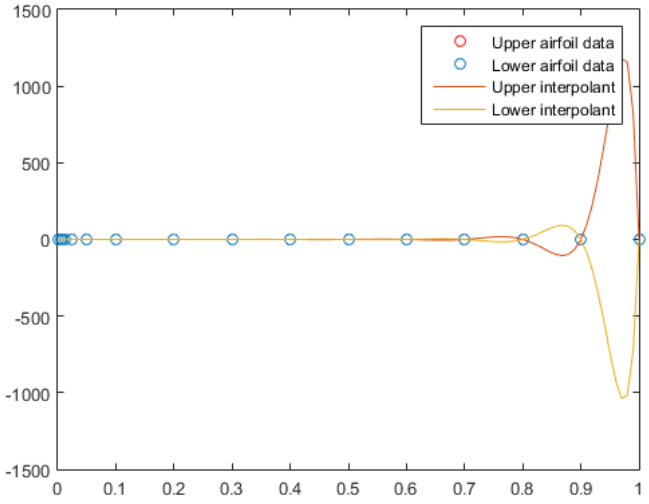

Figure 10: Initial graph of interpolating polynomial

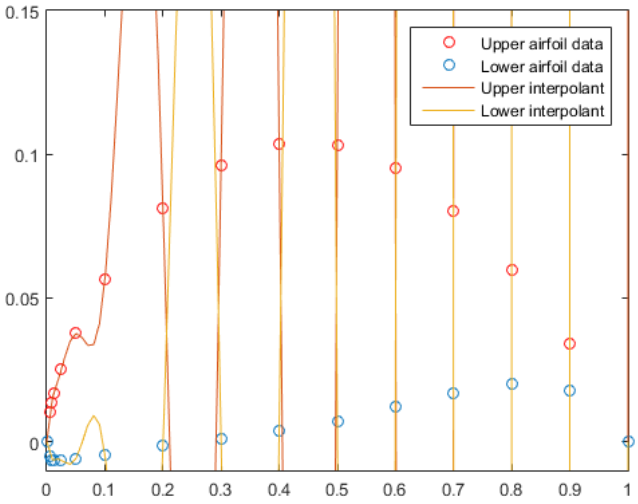

Figure 11: Resized graph of interpolating polynomial

6. In order to perform all the necessary calculations on the data, a MATLAB script `q6.m` was used. This script defines the given data points, and uses the built-in routine `fit` to generate a function of the form

$$f(x) = c_1 t e^{c_2 t}$$

This function is used to calculate the range of times in which the drug can operate correctly, i.e. when the concentration is higher than 4 ng/ml. The results of the calculations are plotted in Figure 12. The graph shows the fitting function $f(x)$ in an extended range in order to show the effective period of the drug, as well as the initial data, and the point at which 50% of the drug is present in the bloodstream.

The script produces a short output, displaying the maximum concentration, as well as the time interval in which the drug is effective.

$$\text{Maximum concentration} = 14.18872$$
$$\text{Effective time range (hours)} = [0.63711, 19.28087]$$

`q6.m` script

```
% fit data regarding bloodstream concentration

% time (hr)
x = (1:1:10)';

% data points
y = [6.2 9.5 12.3 13.9 14.6 13.5 13.3 12.7 12.4 11.9]';

% use 'fit' routine to fit data to model
ff = fit(x,y,'a*x*exp(b*x)');

% save coefficients
c = coeffvalues(ff);

% explicitly define the function anonymously, since
% cfit objects aren't as nice to work with
f =@(x) c(1)*x*exp(c(2)*x);

% generate vector of values
xx = linspace(1,10);
fv = feval(ff,xx);

% find maximum concentration
maxy = max(fv);
fprintf('Maximum fit concentration level: %.5f\n', maxy);

% calculate zeros of f-4. t values in between these zeros are the
% times during which the drug is within therapeutic range
fz = @(x) c(1)*x*exp(c(2)*x)-4;
z1 = fzero(fz, 10);
z2 = fzero(fz,50);

fprintf('The concentration will be outside the therapeutic range from ');
```

```
fprintf('t = %.5f to t=%.5f\n',z1,z2);

% calculate 50% concentration line
xh = linspace(0,25);
hy = maxy/2+0.*xh;

hold off
% plot fitting function, color according to therapeutic range
fplot(f,[z1 z2],'g');
hold all

% plot initial data
plot(x,y,'*');

% plot 50% concentration line
plot(xh,hy);

% plot f(x) that is outside therapeutic range
fplot(f,[z2 25],'r');
fplot(f,[0 z1],'r');

% add information
ylim([0 20]);
xlim([0 25]);
xlabel('t (hr)');    ylabel('y (ng/ml)');
legend('f(x)','data','50% max concentration','f(x) outside therapeutic range');
```

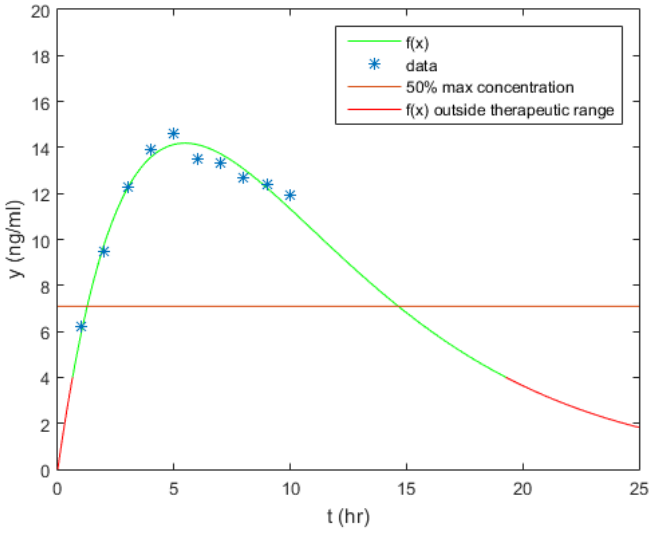

Figure 12: Graph of $f(x)$, given data, and 50% concentration line

7. The `profile.m` script is used in this problem to calculate a parametric cubic spline to form a cursive 'A'. The script takes as input a set of points specified by the user, and performes 3 spline interpolations. The first two are fitting splines to the values of x and y versus i, i being the index of the coordinate point. There was one slight modification made to the code, which was the use of an external program "GeoGebra" to more accurately select points. Every attempt made in MATLAB to choose well-fitting points resulted in a spline barely resembling an 'A'. GeoGebra simply allows points to be selected and modified, so a much cleaner looking result could be achieved by `profile.m`.

`profile.m` script

```
% script to generate a picture from given data points
clc;
close all;

% generate data points - this was done using GeoGebra in order to get a
% nicer picture
x = [0.83; 0.76; 0.67; 0.51; 0.41; 0.33; 0.27; 0.27; 0.35; 0.47; 0.59; 0.71;
    0.83; 0.91; 0.93; 0.92; 0.89; 0.84; 0.82; 0.83; 0.86; 0.89; 0.93; 0.97; ];

y = [0.75; 0.78; 0.81; 0.81; 0.74; 0.63; 0.47; 0.30; 0.21; 0.17; 0.15; 0.17;
    0.24; 0.35; 0.50; 0.67; 0.74; 0.66; 0.50; 0.41; 0.32; 0.25; 0.18; 0.12; ];

% keep track of the total number of data points
n = length(x);

% display data results
fprintf('Initial data\n');
fprintf('     i      x_i          y_i\n');

for ii = 1:n
    fprintf('     %d      %.5f      %.5f\n',ii,x(ii),y(ii));
end

% define constant spacing
i = (1:1:n);

% compute splines for x and y values
sx = spline(i,x);
sy = spline(i,y);

% generate vectors for domain, x data, y data
d = linspace(1,n);
zx = ppval(sx,d);
zy = ppval(sy,d);

% plot x spline and data points
plot(i,x,'*',d,zx);
xlabel('i');
```

```
ylabel('x');

% plot y spline and data points
figure;
plot(i,y,'*',d,zy);
xlabel('i');
ylabel('y');

% plot full spline and data points
figure;
plot(zx,zy,x,y,'*');
```

`profile.m` output

```
Initial data
     i      x_i          y_i
     1      0.83000      0.75000
     2      0.76000      0.78000
     3      0.67000      0.81000
     4      0.51000      0.81000
     5      0.41000      0.74000
     6      0.33000      0.63000
     7      0.27000      0.47000
     8      0.27000      0.30000
     9      0.35000      0.21000
     10      0.47000      0.17000
     11      0.59000      0.15000
     12      0.71000      0.17000
     13      0.83000      0.24000
     14      0.91000      0.35000
     15      0.93000      0.50000
     16      0.92000      0.67000
     17      0.89000      0.74000
     18      0.84000      0.66000
     19      0.82000      0.50000
     20      0.83000      0.41000
     21      0.86000      0.32000
     22      0.89000      0.25000
     23      0.93000      0.18000
     24      0.97000      0.12000
```

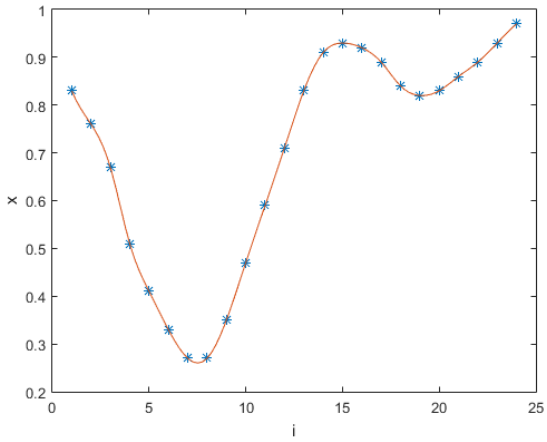

Figure 13: Graph of x-coordinate spline

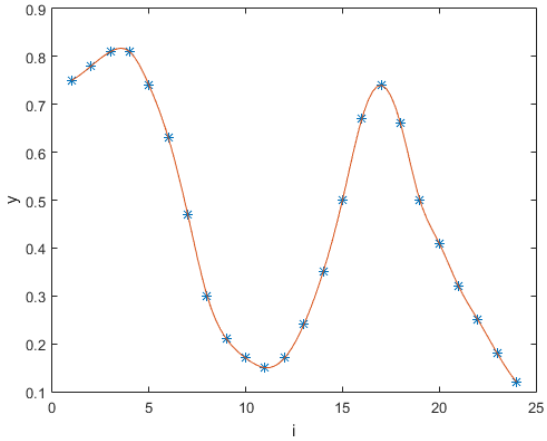

Figure 14: Graph of y-coordinate spline

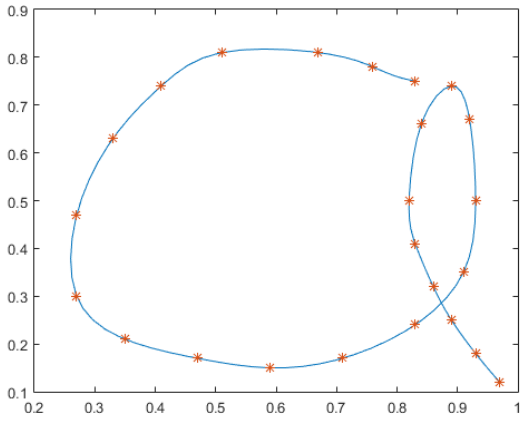

Figure 15: Graph of full parametric spline